# NAG Toolbox

# nag_wav_3d_multi_fwd (c09fc)

## 1 Purpose

nag_wav_3d_multi_fwd (c09fc) computes the three-dimensional multi-level discrete wavelet transform (DWT). The initialization function nag_wav_3d_init (c09ac) must be called first to set up the DWT options.

## 2 Syntax

```
[c, dwtlvm, dwtlvn, dwtlvfr, icomm, ifail] = nag_wav_3d_multi_fwd(n, fr, a, lenc, nwl, icomm, 'm', m)
[c, dwtlvm, dwtlvn, dwtlvfr, icomm, ifail] = c09fc(n, fr, a, lenc, nwl, icomm, 'm', m)
```

## 3 Description

nag_wav_3d_multi_fwd (c09fc) computes the multi-level DWT of three-dimensional data. For a given wavelet and end extension method, nag_wav_3d_multi_fwd (c09fc) will compute a multi-level transform of a three-dimensional array $A$, using a specified number, $n_{\mathrm{fwd}}$, of levels. The number of levels specified, $n_{\mathrm{fwd}}$, must be no more than the value $l_{\mathrm{max}}$ returned in **nwlmax** by the initialization function nag_wav_3d_init (c09ac) for the given problem. The transform is returned as a set of coefficients for the different levels (packed into a single array) and a representation of the multi-level structure.

The notation used here assigns level 0 to the input data, $A$. Level 1 consists of the first set of coefficients computed: the seven sets of detail coefficients are stored at this level while the approximation coefficients are used as the input to a repeat of the wavelet transform at the next level. This process is continued until, at level $n_{\mathrm{fwd}}$, all eight types of coefficients are stored. All coefficients are packed into a single array.

## 4 References

Wang Y, Che X and Ma S (2012) Nonlinear filtering based on 3D wavelet transform for MRI denoising *URASIP Journal on Advances in Signal Processing* **2012:40**

## 5 Parameters

### 5.1 Compulsory Input Parameters

1: **n** – INTEGER

The number of columns of each two-dimensional frame.

*Constraint*: this must be the same as the value **n** passed to the initialization function nag_wav_3d_init (c09ac).

2: **fr** – INTEGER

The number of two-dimensional frames.

*Constraint*: this must be the same as the value **fr** passed to the initialization function nag_wav_3d_init (c09ac).

3: **a**$(lda, sda, \mathbf{fr})$ – REAL (KIND=nag_wp) array

$lda$, the first dimension of the array, must satisfy the constraint $lda \geq \mathbf{m}$.

The $m$ by $n$ by $fr$ three-dimensional input data $A$, where with $A_{ijk}$ stored in $\mathbf{a}(i, j, k)$.

4: **lenc** – INTEGER

The dimension of the array **c**.

*Constraint*: **lenc** $\geq n_{\text{ct}}$, where $n_{\text{ct}}$ is the total number of wavelet coefficients that correspond to a transform with **nwl** levels.

5: **nwl** – INTEGER

The number of levels, $n_{\text{fwd}}$, in the multi-level resolution to be performed.

*Constraint*: $1 \leq \mathbf{nwl} \leq l_{\text{max}}$, where $l_{\text{max}}$ is the value returned in **nwlmax** (the maximum number of levels) by the call to the initialization function nag_wav_3d_init (c09ac).

6: **icomm**(**260**) – INTEGER array

Contains details of the discrete wavelet transform and the problem dimension as setup in the call to the initialization function nag_wav_3d_init (c09ac).

## 5.2 Optional Input Parameters

1: **m** – INTEGER

*Default*: the first dimension of the array **a**.

The number of rows of each two-dimensional frame.

*Constraint*: this must be the same as the value **m** passed to the initialization function nag_wav_3d_init (c09ac).

## 5.3 Output Parameters

1: **c**(**lenc**) – REAL (KIND=nag_wp) array

The coefficients of the discrete wavelet transform. If you need to access or modify the approximation coefficients or any specific set of detail coefficients then the use of nag_wav_3d_coeff_ext (c09fy) or nag_wav_3d_coeff_ins (c09fz) is recommended. For completeness the following description provides details of precisely how the coefficients are stored in **c** but this information should only be required in rare cases.

Let $q(i)$ denote the number of coefficients of each type at level $i$, for $i = 1, 2, \ldots, n_{\text{fwd}}$, such that $q(i) = \mathbf{dwtlvm}(n_{\text{fwd}} - i + 1) \times \mathbf{dwtlvn}(n_{\text{fwd}} - i + 1) \times \mathbf{dwtlvfr}(n_{\text{fwd}} - i + 1)$. Then, letting $k_1 = q(n_{\text{fwd}})$ and $k_{j+1} = k_j + q(n_{\text{fwd}} - \lceil j/7 \rceil + 1)$, for $j = 1, 2, \ldots, 7n_{\text{fwd}}$, the coefficients are stored in **c** as follows:

**c**$(i)$, for $i = 1, 2, \ldots, k_1$
Contains the level $n_{\text{fwd}}$ approximation coefficients, $a_{n_{\text{fwd}}}$. Note that for computational efficiency reasons these coefficients are stored as $\mathbf{dwtlvm}(1) \times \mathbf{dwtlvn}(1) \times \mathbf{dwtlvfr}(1)$ in **c**.

**c**$(i)$, for $i = k_j + 1, \ldots, k_{j+1}$
Contains the level $n_{\text{fwd}} - \lceil j/7 \rceil + 1$ detail coefficients. These are:

LLH coefficients if $j \bmod 7 = 1$;

LHL coefficients if $j \bmod 7 = 2$;

LHH coefficients if $j \bmod 7 = 3$;

HLL coefficients if $j \bmod 7 = 4$;

HLH coefficients if $j \bmod 7 = 5$;

HHL coefficients if $j \bmod 7 = 6$;

HHH coefficients if $j \bmod 7 = 0$,

for $j = 1, \ldots, 7n_{\text{fwd}}$. See Section 2.1 in the C09 Chapter Introduction for a description of how these coefficients are produced.

Note that for computational efficiency reasons these coefficients are stored as **dwtlvfr**$(\lceil j/7 \rceil) \times$ **dwtlvm**$(\lceil j/7 \rceil) \times$ **dwtlvn**$(\lceil j/7 \rceil)$ in **c**.

2: **dwtlvm**(**nwl**) – INTEGER array

The number of coefficients in the first dimension for each coefficient type at each level. **dwtlvm**$(i)$ contains the number of coefficients in the first dimension (for each coefficient type computed) at the $(n_{\text{fwd}} - i + 1)$th level of resolution, for $i = 1, 2, \ldots, n_{\text{fwd}}$.

3: **dwtlvn**(**nwl**) – INTEGER array

The number of coefficients in the second dimension for each coefficient type at each level. **dwtlvn**$(i)$ contains the number of coefficients in the second dimension (for each coefficient type computed) at the $(n_{\text{fwd}} - i + 1)$th level of resolution, for $i = 1, 2, \ldots, n_{\text{fwd}}$.

4: **dwtlvfr**(**nwl**) – INTEGER array

The number of coefficients in the third dimension for each coefficient type at each level. **dwtlvfr**$(i)$ contains the number of coefficients in the third dimension (for each coefficient type computed) at the $(n_{\text{fwd}} - i + 1)$th level of resolution, for $i = 1, 2, \ldots, n_{\text{fwd}}$.

5: **icomm**(**260**) – INTEGER array

Contains additional information on the computed transform.

6: **ifail** – INTEGER

**ifail** $= 0$ unless the function detects an error (see Section 5).

## 6 Error Indicators and Warnings

Errors or warnings detected by the function:

**ifail** $= 1$

Constraint: **fr** $= \langle value \rangle$, the value of **fr** on initialization (see nag_wav_3d_init (c09ac)).

Constraint: **m** $= \langle value \rangle$, the value of **m** on initialization (see nag_wav_3d_init (c09ac)).

Constraint: **n** $= \langle value \rangle$, the value of **n** on initialization (see nag_wav_3d_init (c09ac)).

**ifail** $= 2$

Constraint: $lda \geq$ **m**.

Constraint: $sda \geq$ **n**.

**ifail** $= 3$

**lenc** is too small, the total number of coefficents to be generated.

**ifail** $= 5$

Constraint: **nwl** $\leq$ **nwlmax** in nag_wav_3d_init (c09ac).

Constraint: **nwl** $\geq 1$.

**ifail** $= 6$

Either the communication array **icomm** has been corrupted or there has not been a prior call to the initialization function nag_wav_3d_init (c09ac).

The initialization function was called with **wtrans** = 'S'.

**ifail** $= -99$

An unexpected error has been triggered by this routine. Please contact NAG.

**ifail** $= -399$

Your licence key may have expired or may not have been installed correctly.

**ifail** $= -999$

Dynamic memory allocation failed.

## 7 Accuracy

The accuracy of the wavelet transform depends only on the floating-point operations used in the convolution and downsampling and should thus be close to ***machine precision***.

## 8 Further Comments

The example program shows how the wavelet coefficients at each level can be extracted from the output array **c**. Denoising can be carried out by applying a thresholding operation to the detail coefficients at every level. If $c_{ij}$ is a detail coefficient then $\hat{c}_{ij} = c_{ij} + \sigma\epsilon_{ij}$ and $\sigma\epsilon_{ij}$ is the transformed noise term. If some threshold parameter $\alpha$ is chosen, a simple hard thresholding rule can be applied as

$$\bar{c}_{ij} = \begin{cases} 0, & \text{if } \left|\hat{c}_{ij}\right| \leq \alpha \\ \hat{c}_{ij}, & \text{if } \left|\hat{c}_{ij}\right| > \alpha, \end{cases}$$

taking $\bar{c}_{ij}$ to be an approximation to the required detail coefficient without noise, $c_{ij}$. The resulting coefficients can then be used as input to nag_wav_3d_mxolap_multi_inv (c09fd) in order to reconstruct the denoised signal. See Section 10 in nag_wav_3d_coeff_ins (c09fz) for a simple example of denoising.

See the references given in the introduction to this chapter for a more complete account of wavelet denoising and other applications.

## 9 Example

This example computes the three-dimensional multi-level discrete wavelet decomposition for $7 \times 6 \times 5$ input data using the biorthogonal wavelet of order 1.1 (set **wavnam** = 'BIOR1.1' in nag_wav_3d_init (c09ac)) with periodic end extension, prints a selected set of wavelet coefficients and then reconstructs and verifies that the reconstruction matches the original data.

### 9.1 Program Text

```
function c09fc_example

fprintf('c09fc example results\n\n');

% Data
m  = nag_int(7);
n  = nag_int(6);
fr = nag_int(5);
a = zeros(m, n, fr);
a(:, :, 1) = [3, 2, 2, 2, 1, 1;
              2, 9, 1, 2, 1, 3;
              2, 5, 1, 2, 1, 1;
              1, 6, 2, 2, 7, 2;
```

```
              5, 3, 2, 2, 4, 7;
              2, 2, 1, 1, 2, 1;
              6, 2, 1, 3, 6, 9];
a(:, :, 2) = [2, 1, 5, 1, 2, 3;
              2, 9, 5, 2, 1, 2;
              2, 3, 2, 7, 1, 1;
              2, 1, 1, 2, 3, 1;
              2, 1, 2, 8, 3, 3;
              1, 4, 5, 1, 2, 7;
              8, 1, 3, 9, 1, 2];
a(:, :, 3) = [3, 1, 4, 1, 1, 1;
              1, 1, 2, 1, 2, 6;
              4, 1, 7, 2, 5, 6;
              3, 2, 1, 5, 9, 5;
              1, 1, 2, 2, 2, 1;
              2, 6, 3, 9, 5, 1;
              1, 1, 8, 2, 1, 3];
a(:, :, 4) = [5, 8, 1, 2, 2, 1;
              1, 2, 2, 9, 2, 9;
              2, 2, 2, 1, 1, 3;
              1, 1, 1, 5, 1, 2;
              3, 2, 8, 1, 9, 2;
              2, 1, 9, 1, 2, 2;
              3, 6, 5, 3, 2, 2];
a(:, :, 5) = [5, 2, 1, 2, 1, 1;
              3, 1, 9, 1, 2, 1;
              2, 3, 1, 1, 7, 2;
              7, 2, 2, 6, 1, 1;
              5, 1, 7, 2, 1, 1;
              2, 1, 3, 2, 2, 1;
              5, 3, 9, 1, 4, 1];

% Query wavelet filter dimensions
wavnam = 'Bior1.1';
mode   = 'period';
wtrans = 'Multilevel';
[nwl, nf, nwct, nwcn, nwcfr, icomm, ifail] = ...
  c09ac(...
         wavnam, wtrans, mode, m, n, fr);

% Perform Discrete Wavelet transform
[c, dwtlvm, dwtlvn, dwtlvfr, icomm, ifail] = ...
  c09fc(...
         n, fr, a, nwct, nwl, icomm);

fprintf(' Number of Levels : %d\n\n', nwl);
fprintf(' Number of coefficients in 1st dimension for each level:\n');
fprintf(' %8d', dwtlvm(1:nwl));
fprintf('\n');
fprintf(' Number of coefficients in 2nd dimension for each level:\n');
fprintf(' %8d', dwtlvn(1:nwl));
fprintf('\n');
fprintf(' Number of coefficients in 3rd dimension for each level:\n');
fprintf(' %8d', dwtlvfr(1:nwl));
fprintf('\n');

% Print the first level HLL detail coefficients
want_level = nag_int(1);
want_coeffs = nag_int(4);

cpass = char('LLH','LHL','LHH','HLL','HLH','HHL','HHH');

if (want_coeffs==0)
  title = 'Approximation coefficients (LLL)'
else
  title = sprintf('Detail coefficients (%s)',cpass(want_coeffs,:));
end

% Extract coefficients
[d, icomm, ifail] = c09fy(...
                          want_level, want_coeffs, c, icomm);
```

```
fprintf('\n%s\n Level %2d, Coefficients %2d :\n',title, want_level, ...
        want_coeffs );

% Matrix printing arguments
matrix = 'General'; diag   = 'Non-unit'; fmt = 'F9.4';
labrow = 'Integer'; labcol = labrow;
rlabs  = {' '};     clabs  = rlabs;
ncols  = nag_int(80); indent = nag_int(0);

nk = dwtlvfr(nwl-want_level+1);
for k = 1:nk
  title = sprintf('Frame: %3d',k);
  [ifail] =  x04cb(...
                   matrix, diag, d(:,:,k), fmt, title, labrow, ...
                  rlabs, labcol, clabs, ncols, indent);
end

% Reconstruct original data
[b, ifail] = c09fd(nwl, c, m, n, fr, icomm);

% Check reconstruction matches original
eps = 10*double(m*n*fr)*x02aj;
err = a-b;
frob = 0;
for i=1:fr
  fnew = sqrt(sum(sum(err(:,:,i).^2)));
  frob = max(frob,fnew);
end

if frob < eps
  fprintf('\nSuccess: the reconstruction matches the original.\n');
else
  fprintf('\nFail: Frobenius norm of b-a is too large.\n');
end
```

## 9.2 Program Results

```
     c09fc example results

 Number of Levels : 2

 Number of coefficients in 1st dimension for each level:
        2        4
 Number of coefficients in 2nd dimension for each level:
        2        3
 Number of coefficients in 3rd dimension for each level:
        2        3

Detail coefficients (HLL)
 Level  1, Coefficients  4 :
 Frame:   1
           1        2        3
 1   -4.9497   0.0000   0.0000
 2    0.7071   1.7678  -3.1820
 3    0.7071   2.1213   1.7678
 4    0.0000   0.0000   0.0000
 Frame:   2
           1        2        3
 1    4.2426  -2.1213  -4.9497
 2    0.7071  -0.0000  -0.7071
 3   -1.4142  -3.1820   1.4142
 4    0.0000   0.0000   0.0000
 Frame:   3
           1        2        3
 1    2.1213  -4.9497  -0.7071
```

```
 2   -2.8284  -4.2426   4.9497
 3    2.1213   2.8284  -0.7071
 4    0.0000   0.0000   0.0000

Success: the reconstruction matches the original.
```